قبل از یادگیری عمیق ، مهندسی ویژگی ها مهم ترین بخش روند کار یادگیری ماشین بود ، زیرا الگوریتم‌های سطحی کلاسیک فضای فرضی غنی‌ای برای یادگیری ویژگی‌های مفید به‌طور خودکار نداشتند. نحوه‌ی ارائه داده‌ها به الگوریتم برای موفقیت آن حیاتی بود. به عنوان مثال، قبل از اینکه شبکه‌های عصبی کانولوشنی (CNN) در حل مسأله‌ی طبقه‌بندی ارقام MNIST موفق شوند، راه‌حل‌ها معمولاً بر اساس ویژگی‌های سخت‌کد شده‌ای مانند تعداد حلقه‌ها در تصویر یک عدد، ارتفاع هر عدد در تصویر، هیستوگرام مقادیر پیکسل‌ها و غیره بودند.  
  
 خوشبختانه، یادگیری عمیق مدرن نیاز به بیشتر مهندسی ویژگی‌ها را از بین می‌برد ، زیرا شبکه‌های عصبی قادرند ویژگی‌های مفید را به‌طور خودکار از داده‌های خام استخراج کنند. آیا این بدین معناست که شما دیگر نیازی به نگرانی در مورد مهندسی ویژگی‌ها ندارید تا زمانی که از شبکه‌های عصبی عمیق استفاده می‌کنید؟ خیر، به دو دلیل:   
  
1. **ویژگی‌های خوب هنوز به شما این امکان را می‌دهند که مشکلات را به‌طور شیک‌تر و با استفاده از منابع کمتر حل کنید. به عنوان مثال، حل مسأله خواندن ساعت با استفاده از یک شبکه عصبی کانولوشنی احمقانه خواهد بود.  
  
2. ویژگی‌های خوب به شما این امکان را می‌دهند که یک مشکل را با داده‌های کمتری حل کنید. توانایی مدل‌های یادگیری عمیق برای یادگیری ویژگی‌ها به‌طور خودکار بستگی به داشتن داده‌های آموزشی زیاد دارد؛ اگر تنها چند نمونه داشته باشید، ارزش اطلاعاتی ویژگی‌های آن‌ها بسیار حیاتی می‌شود.  
  
5.4.3 استفاده از توقف زودهنگام**  
  
در یادگیری عمیق، ما همیشه از مدل‌هایی استفاده می‌کنیم که پارامترهای زیادی دارند : این مدل‌ها آزادی‌های بیشتری از حداقل مورد نیاز برای تطبیق با منیفولد (چندوجهی) نهفته داده‌ها دارند. این اضافه‌پارامترسازی مشکلی ایجاد نمی‌کند، زیرا شما هیچ‌گاه یک مدل یادگیری عمیق را به‌طور کامل تطبیق نمی‌دهید. چنین تطبیقی اصلاً قابل تعمیم نخواهد بود. شما همیشه آموزش را خیلی زودتر از رسیدن به حداقل ممکن از دست دادن آموزش متوقف می‌کنید.

یافتن نقطه‌ای که دقیقاً در طول آموزش به بهترین تطبیق تعمیم‌پذیر دست یافته‌اید—حد مرزی دقیق بین منحنی زیرآموزش و منحنی بیش‌ازحد آموزش—یکی از مؤثرترین کارهایی است که می‌توانید برای بهبود تعمیم انجام دهید.

در مثال‌های فصل قبل، ما ابتدا مدل‌های خود را بیشتر از آنچه نیاز است آموزش می‌دهیم تا تعداد اپوک‌هایی که بهترین معیارهای اعتبارسنجی را به دست می‌دهد، پیدا کنیم، سپس مدل جدیدی را دقیقاً برای همان تعداد اپوک‌ها دوباره آموزش می‌دهیم. این روش نسبتاً استاندارد است، اما نیازمند انجام کارهای تکراری است که گاهی اوقات ممکن است هزینه ‌بر باشد. به‌طور طبیعی ، شما می‌توانید مدل خود را در پایان هر اپوک ذخیره کنید ، و زمانی که بهترین اپوک را پیدا کردید ، مدل ذخیره‌ شده‌ای که به آن نزدیک ‌تر است را دوباره استفاده کنید.

در Keras(کتابخانه برای ساخت مدل های Deep Learning)، معمولاً این کار با استفاده از callback توقف زودهنگام (Early Stopping) انجام می‌شود، که به محض اینکه معیارهای اعتبارسنجی بهبود پیدا نکردند، آموزش را متوقف می‌کند و بهترین وضعیت مدل شناخته شده را به یاد می‌سپارد. شما در فصل 7 یاد خواهید گرفت که چگونه از callback‌ها استفاده کنید.  
  
 5.4.4 منظم‌سازی مدل   
 **تنظیم مدل** شامل مجموعه‌ای از بهترین روش‌ها است که به طور فعال مانع از تطبیق کامل مدل با داده‌های آموزشی می‌شوند، با هدف بهبود عملکرد مدل در اعتبارسنجی. این فرآیند به این دلیل "تنظیم" نامیده می‌شود که مدل را ساده‌تر، منظم‌تر و با منحنی نرم‌تر می‌سازد. نتیجه آن است که مدل به جای اینکه بیش از حد به داده‌های آموزشی وابسته باشد، بهتر می‌تواند تعمیم دهد و ساختار پنهان داده‌ها را بهتر تخمین بزند.

به خاطر داشته باشید که تنظیم مدل فرآیندی است که همواره باید با استفاده از یک روش ارزیابی دقیق هدایت شود. شما تنها زمانی به قابلیت تعمیم دست پیدا می‌کنید که بتوانید آن را اندازه‌گیری کنید.

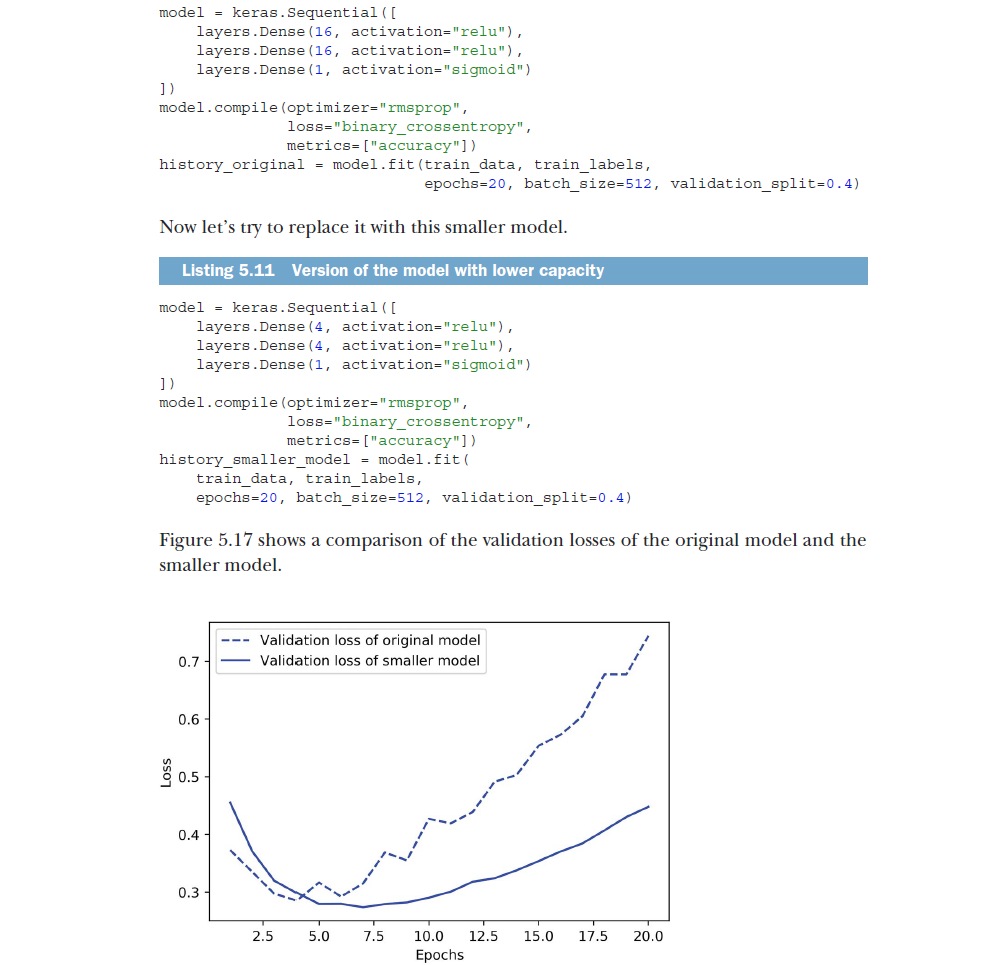
بیایید برخی از تکنیک‌های تنظیم رایج را مرور کرده و آن‌ها را به صورت عملی برای بهبود مدل دسته‌بندی فیلم‌ها از فصل ۴ استفاده کنیم.

**کاهش اندازه‌ی شبکه**

پیش‌تر یاد گرفتید که مدلی که بسیار کوچک باشد، بیش‌تطبیق نمی‌کند. ساده‌ترین راه برای کاهش بیش‌تطبیق، کاهش اندازه‌ی مدل است (یعنی تعداد پارامترهای قابل یادگیری در مدل که با تعداد لایه‌ها و تعداد واحدها در هر لایه تعیین می‌شوند). اگر مدل منابع حافظه‌ای محدودی داشته باشد، نمی‌تواند داده‌های آموزشی خود را حفظ کند. بنابراین، برای به حداقل رساندن خطای خود مجبور خواهد شد نمایش‌های فشرده‌ای را یاد بگیرد که قدرت پیش‌بینی در مورد اهداف دارند—دقیقاً همان نوع نمایش‌هایی که به دنبال آن هستیم.

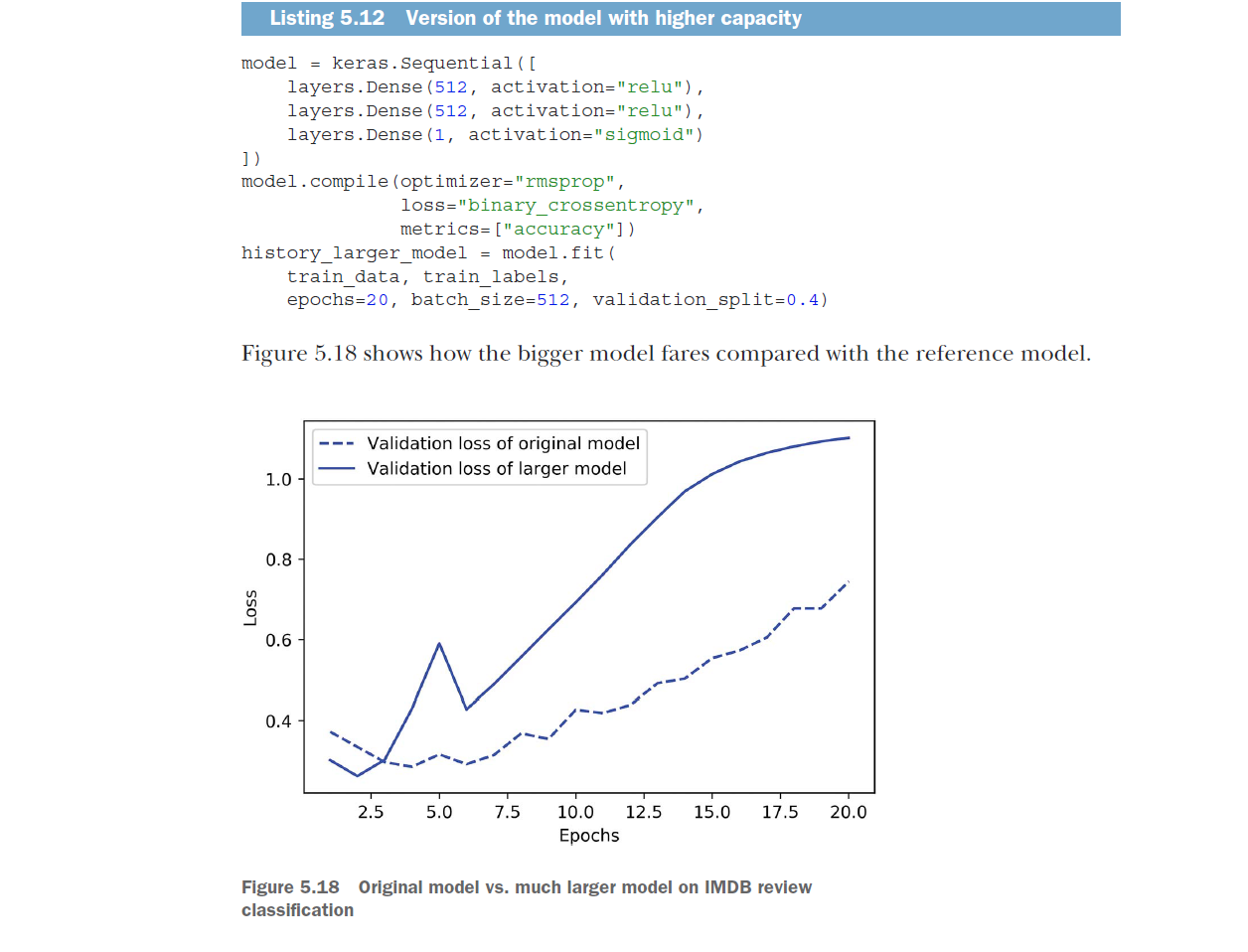
با این حال، به خاطر داشته باشید که باید از مدل‌هایی استفاده کنید که به اندازه‌ی کافی پارامتر داشته باشند تا دچار کم‌تطبیق نشوند. نباید منابع حافظه‌ای مدل را محدود کنید. یک نقطه‌ی تعادل بین ظرفیت بیش از حد و ظرفیت ناکافی وجود دارد که باید پیدا شود.

متأسفانه، هیچ فرمول جادویی برای تعیین تعداد لایه‌های مناسب یا اندازه‌ی هر لایه وجود ندارد. شما باید معماری‌های مختلف را (روی مجموعه‌ی اعتبارسنجی، نه مجموعه‌ی آزمایش) ارزیابی کنید تا اندازه‌ی مناسب مدل برای داده‌های خود را پیدا کنید.

A screenshot of a computer program

Description automatically generated

همان‌طور که مشاهده می‌کنید، مدل کوچک‌تر نسبت به مدل مرجع دیرتر دچار بیش‌تطبیق می‌شود (پس از شش اپوک به جای چهار اپوک) و عملکرد آن پس از آغاز بیش‌تطبیق، به‌تدریج و با سرعت کمتری کاهش می‌یابد.

حالا، بیایید مدلی را به معیارهای خود اضافه کنیم که ظرفیت بسیار بیشتری دارد—بسیار بیشتر از آنچه مسئله نیاز دارد. اگرچه استفاده از مدل‌هایی با بیش‌پارامتری قابل توجه برای یادگیری مسائل استاندارد است، اما به طور قطع ظرفیت حافظه‌ی بیش از حد نیز ممکن است وجود داشته باشد. شما متوجه خواهید شد که مدل‌تان بیش از حد بزرگ است اگر بلافاصله دچار بیش‌تطبیق شود و منحنی خطای اعتبارسنجی آن دارای نوسانات زیاد و تغییرات بالا باشد (البته نوسانات در معیارهای اعتبارسنجی همچنین ممکن است نشانه‌ی استفاده از یک فرآیند اعتبارسنجی غیرقابل اعتماد باشد، مانند تقسیم‌بندی بسیار کوچک برای مجموعه‌ی اعتبارسنجی).

مدل بزرگ‌تر تقریباً بلافاصله، پس از تنها یک اپوک، دچار بیش‌تطبیق می‌شود و این بیش‌تطبیق به شکل بسیار شدیدتری رخ می‌دهد. **خطای اعتبارسنجی آن نیز نوسان بیشتری دارد**. این مدل خیلی سریع به خطای آموزشی نزدیک به صفر می‌رسد. هرچه ظرفیت مدل بیشتر باشد، سریع‌تر می‌تواند داده‌های آموزشی را مدل‌سازی کند (که به کاهش خطای آموزشی منجر می‌شود)، اما به همان نسبت مستعد بیش‌تطبیق می‌شود (که به اختلاف زیاد بین خطای آموزشی و خطای اعتبارسنجی می‌انجامد).

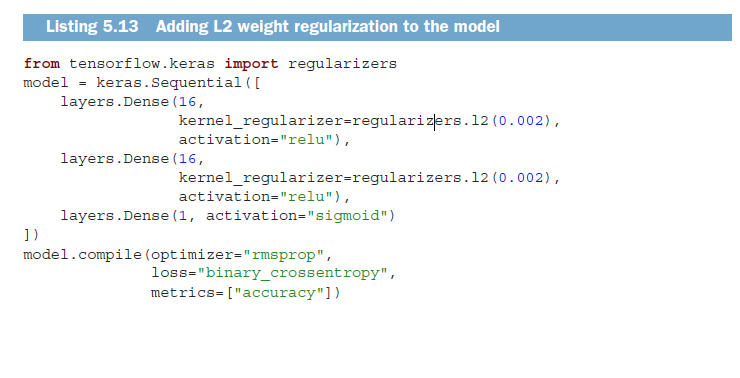
**افزودن تنظیم وزن‌ها (Weight Regularization)**

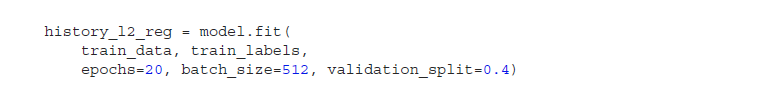
ممکن است با اصل "تیغ اوکام" آشنا باشید: با فرض دو توضیح برای یک موضوع، توضیحی که ساده‌تر باشد و فرضیات کمتری را شامل شود، به احتمال زیاد صحیح‌تر است. این ایده در مورد مدل‌هایی که توسط شبکه‌های عصبی یاد گرفته می‌شوند نیز صدق می‌کند: با داشتن مجموعه‌ای از داده‌های آموزشی و یک معماری شبکه، چندین مجموعه از مقادیر وزن (چندین مدل) می‌توانند داده‌ها را توضیح دهند. مدل‌های ساده‌تر کمتر احتمال دارد که دچار بیش‌تطبیق شوند، در مقایسه با مدل‌های پیچیده‌تر.

در این زمینه، یک مدل ساده مدلی است که توزیع مقادیر پارامترهای آن آنتروپی کمتری دارد (یا مدلی که تعداد پارامترهای کمتری دارد، همان‌طور که در بخش قبلی دیدید). بنابراین، یک روش رایج برای کاهش بیش‌تطبیق، اعمال محدودیت بر پیچیدگی مدل است؛ به این صورت که وزن‌های مدل را مجبور به داشتن مقادیر کوچک می‌کنیم، که باعث می‌شود توزیع وزن‌ها منظم‌تر باشد. این فرآیند به **تنظیم وزن‌ها** معروف است و با اضافه کردن یک هزینه به تابع خطای مدل انجام می‌شود که با داشتن وزن‌های بزرگ مرتبط است. این هزینه به دو شکل اعمال می‌شود:

1. **تنظیم L1:** هزینه اضافه‌شده متناسب با مقدار مطلق ضرایب وزنی است (L1 norm).
2. **تنظیم L2:** هزینه اضافه‌شده متناسب با مربع مقدار ضرایب وزنی است (L2 norm). تنظیم L2 در زمینه شبکه‌های عصبی به عنوان **کاهش وزن (weight decay)** نیز شناخته می‌شود. اجازه ندهید نام‌های متفاوت شما را گیج کند؛ کاهش وزن به لحاظ ریاضی همان تنظیم L2 است.

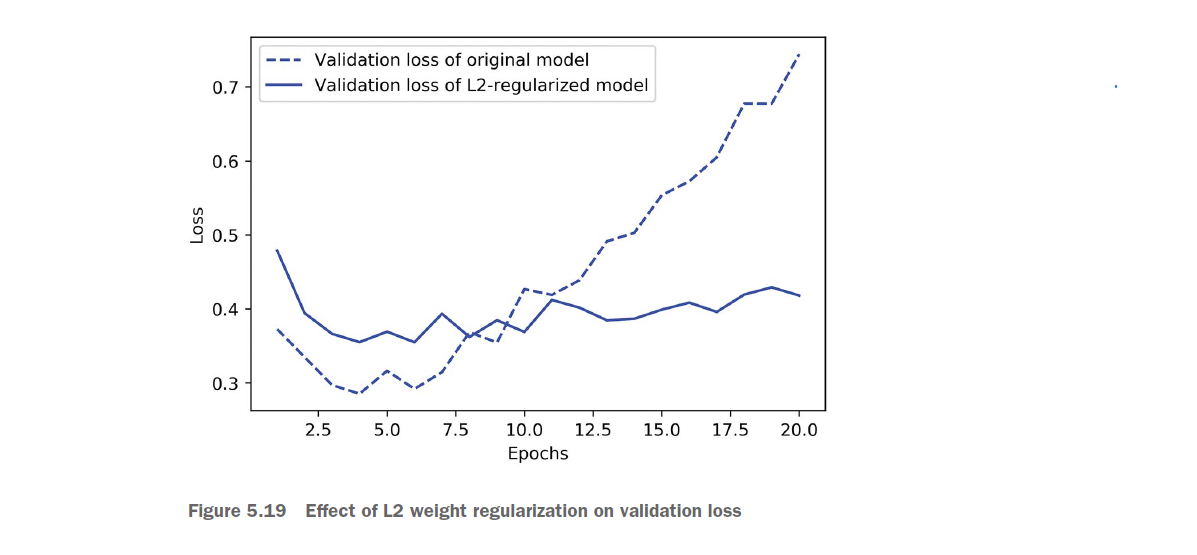
در Keras، تنظیم وزن‌ها با انتقال نمونه‌هایی از تنظیم‌کننده وزن به لایه‌ها به عنوان آرگومان‌های کلیدواژه‌ای انجام می‌شود. بیایید تنظیم وزن L2 را به مدل اولیه‌ی دسته‌بندی نظرات فیلم اضافه کنیم.





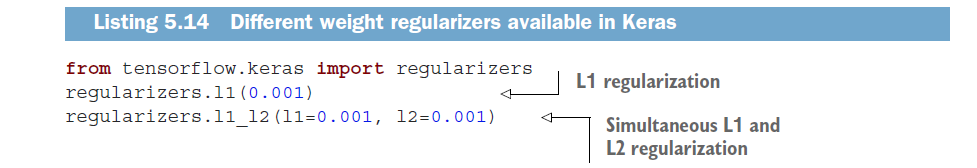
در لیست بالا، مقدار l2(0.002) به این معناست که هر ضریب در ماتریس وزن لایه ،  
 مقدار 0.002 \* weight\_coefficient\_value \*\* 2 را به مجموع خطای مدل اضافه خواهد کرد. توجه داشته باشید که این جریمه فقط در زمان آموزش اعمال می‌شود، بنابراین خطای این مدل در زمان آموزش بسیار بیشتر از زمان آزمایش خواهد بود.

**شکل ۵.۱۹** تأثیر جریمه تنظیم L2 را نشان می‌دهد. همان‌طور که مشاهده می‌کنید، مدلی که از تنظیم L2 استفاده می‌کند، در مقایسه با مدل مرجع، مقاومت بیشتری در برابر بیش‌تطبیق پیدا کرده است، حتی با اینکه هر دو مدل تعداد پارامترهای یکسانی دارند.



به‌عنوان جایگزینی برای تنظیم L2، می‌توانید از یکی از تنظیم‌کننده‌های وزن Keras استفاده کنید.

توجه داشته باشید که تنظیم وزن‌ها معمولاً برای مدل‌های کوچک یادگیری عمیق بیشتر استفاده می‌شود. مدل‌های بزرگ یادگیری عمیق معمولاً آن‌قدر بیش‌پارامتری هستند که اعمال محدودیت بر مقادیر وزن‌ها تأثیر زیادی بر ظرفیت مدل و تعمیم آن ندارد. در چنین مواردی، تکنیک تنظیم دیگری ترجیح داده می‌شود: **دراپ‌اوت (Dropout)**.

 **افزودن Dropout**

**Dropout** یکی از مؤثرترین و رایج‌ترین تکنیک‌های تنظیم برای شبکه‌های عصبی است که توسط **جف هینتون** و دانشجویان او در دانشگاه تورنتو توسعه داده شد. **Dropout** هنگامی که به یک لایه اعمال می‌شود، شامل حذف تصادفی (صفر کردن) تعدادی از ویژگی‌های خروجی لایه در زمان آموزش است.

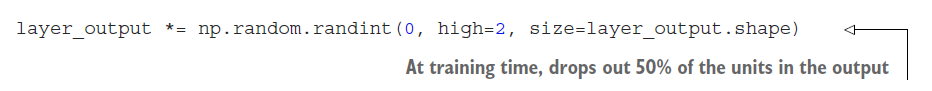
برای مثال، فرض کنید که یک لایه معمولاً برای یک نمونه ورودی، برداری مانند :  
 [0.2, 0.5, 1.3, 0.8, 1.1] تولید می‌کند. پس از اعمال Dropout، این بردار ممکن است شامل چند مقدار صفر به طور تصادفی باشد، مثلاً [0, 0.5, 1.3, 0, 1.1]. **نرخ Dropout** درصدی از ویژگی‌ها است که صفر می‌شوند؛ این نرخ معمولاً بین ۰.۲ تا ۰.۵ تنظیم می‌شود.

در زمان **آزمایش** (Test Time)، هیچ واحدی حذف نمی‌شود؛ بلکه مقادیر خروجی لایه‌ها با یک ضریب برابر با نرخ Dropout مقیاس‌بندی می‌شوند. این کار برای جبران تعداد بیشتری از واحدهای فعال در زمان آزمایش نسبت به زمان آموزش انجام می‌شود.

**توضیح با یک مثال NumPy**

فرض کنید یک ماتریس NumPy به نام layer\_output دارید که شامل خروجی یک لایه با شکل (batch\_size, features) است:

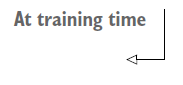
1. **در زمان آموزش**، بخشی از مقادیر ماتریس به صورت تصادفی صفر می‌شود:

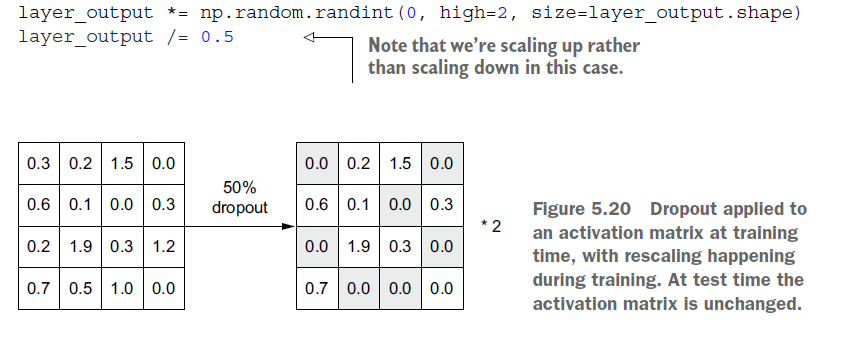


**در زمان آزمایش**، خروجی با نرخ Dropout مقیاس‌بندی می‌شود. برای مثال، اگر نیمی از واحدها قبلاً حذف شده‌اند:



توجه داشته باشید که این فرآیند می‌تواند به طور کامل در زمان آموزش انجام شود و خروجی در زمان آزمایش بدون تغییر باقی بماند. این معمولاً روشی است که در عمل پیاده‌سازی می‌شود (همان‌طور که در شکل ۵.۲۰ نشان داده شده است)



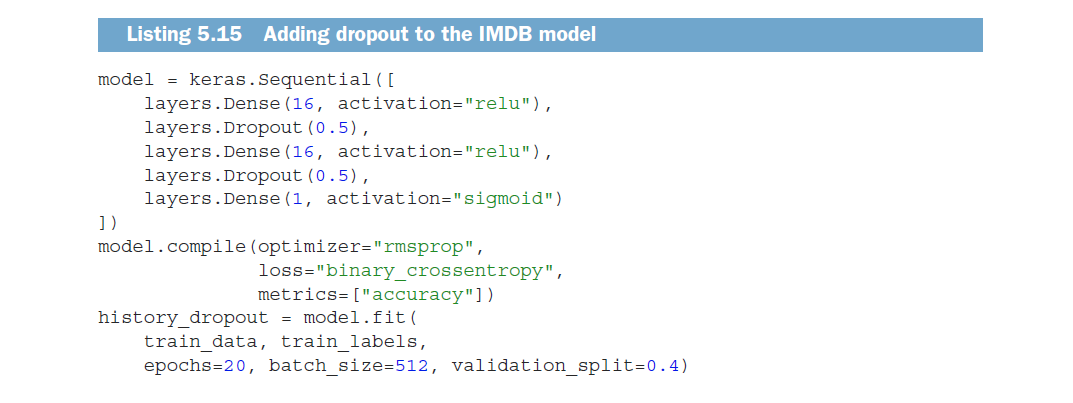


این تکنیک ممکن است عجیب و دل‌بخواه به نظر برسد. چرا چنین روشی می‌تواند به کاهش بیش‌تطبیق کمک کند؟

**هینتون** می‌گوید که از مکانیزم‌های پیشگیری از تقلب در بانک‌ها، در کنار سایر موارد، الهام گرفته است. به گفته‌ی او:  
«به بانکم رفتم. متصدیان همیشه تغییر می‌کردند، و از یکی از آن‌ها پرسیدم چرا. او گفت که نمی‌داند، «اما مرتباً جابه‌جا می‌شوند.   
  
  
  
  
  
  
  
متوجه شدم که احتمالاً دلیل این جابه‌جایی‌ها این است که برای تقلب موفقیت‌آمیز، نیاز به همکاری بین کارمندان وجود دارد. این موضوع باعث شد دریابم که حذف تصادفی زیرمجموعه‌ای از نورون‌ها در هر نمونه، می‌تواند از ایجاد «تبانی‌ها» جلوگیری کرده و در نتیجه، **بیش‌تطبیق** را کاهش دهد.

ایده‌ی اصلی این است که اضافه کردن نویز به مقادیر خروجی یک لایه می‌تواند الگوهای تصادفی و بی‌اهمیت را بشکند (چیزی که هینتون به آن «تبانی‌ها» می‌گوید)؛ الگویی که اگر نویزی وجود نداشته باشد، مدل شروع به حفظ کردن آن‌ها می‌کند.

در کتابخانه‌ی **Keras**، می‌توانید از لایه‌ی **Dropout** برای اعمال این تکنیک در یک مدل استفاده کنید. این لایه به خروجی لایه‌ی قبل از خود اعمال می‌شود. بیایید دو لایه‌ی Dropout به مدل تحلیل نظرات IMDB اضافه کنیم تا ببینیم این روش چقدر در کاهش بیش‌تطبیق مؤثر است.



**شکل ۵.۲۱** نمودار نتایج را نشان می‌دهد. این نتایج به وضوح بهتر از مدل مرجع است و به نظر می‌رسد که به مراتب بهتر از تنظیم L2 عمل می‌کند، زیرا کمترین مقدار خطای اعتبارسنجی که به آن دست یافته‌ایم، بهبود یافته است. A graph with blue lines

Description automatically generated

**جمع‌بندی**

این‌ها رایج‌ترین روش‌ها برای به حداکثر رساندن قابلیت تعمیم و جلوگیری از بیش‌تطبیق در شبکه‌های عصبی هستند:

* افزایش داده‌های آموزشی یا استفاده از داده‌های آموزشی باکیفیت‌تر.
* توسعه‌ی ویژگی‌های بهتر.
* کاهش ظرفیت مدل.
* افزودن تنظیم وزن‌ها (برای مدل‌های کوچک‌تر).
* افزودن Dropout.

**خلاصه**

* هدف از یک مدل یادگیری ماشین **تعمیم** است: عملکرد دقیق بر روی داده‌هایی که هرگز پیش از این دیده نشده‌اند. این کار سخت‌تر از آن است که به نظر می‌رسد.
* یک شبکه عصبی عمیق با یادگیری یک مدل پارامتریک که می‌تواند به طور موفقیت‌آمیز بین نمونه‌های آموزشی **میان‌یابی** کند، به تعمیم دست می‌یابد—چنین مدلی می‌تواند گفته شود که "ساختار پنهان" (latent manifold) داده‌های آموزشی را یاد گرفته است. به همین دلیل، مدل‌های یادگیری عمیق فقط می‌توانند ورودی‌هایی را درک کنند که بسیار نزدیک به آنچه در زمان آموزش دیده‌اند، باشند.
* مشکل اساسی در یادگیری ماشین **تنش بین بهینه‌سازی و تعمیم** است: برای دستیابی به تعمیم، ابتدا باید یک تطبیق خوب با داده‌های آموزشی انجام شود، اما بهبود تطبیق مدل با داده‌های آموزشی در نهایت به ضرر تعمیم خواهد بود. هر یک از بهترین روش‌های یادگیری عمیق با مدیریت این تنش سر و کار دارند.
* توانایی مدل‌های یادگیری عمیق در تعمیم به این دلیل است که می‌توانند تقریب خوبی از ساختار پنهان داده‌های خود بیاموزند و از این طریق بتوانند ورودی‌های جدید را از طریق میان‌یابی درک کنند.
* ضروری است که در طول توسعه‌ی مدل، بتوانید قدرت تعمیم مدل خود را به دقت ارزیابی کنید. شما مجموعه‌ای از روش‌های ارزیابی در اختیار دارید، از اعتبارسنجی نگه‌داشت ساده (Holdout Validation) گرفته تا اعتبارسنجی متقاطع K-fold و اعتبارسنجی متقاطع تکراری با شافلینگ. همیشه یک مجموعه‌ی آزمایشی کاملاً جداگانه برای ارزیابی نهایی مدل نگه دارید، زیرا ممکن است اطلاعات از داده‌های اعتبارسنجی به مدل شما نفوذ کرده باشند.
* هنگامی که کار روی یک مدل را شروع می‌کنید، هدف اولیه‌ی شما این است که مدلی داشته باشید که مقداری قدرت تعمیم داشته باشد و بتواند دچار بیش‌تطبیق شود. بهترین روش‌ها برای این کار شامل تنظیم نرخ یادگیری و اندازه‌ی بچ، استفاده از پیش‌فرض‌های بهتر در معماری، افزایش ظرفیت مدل، یا صرفاً آموزش طولانی‌تر است.
* هنگامی که مدل شما شروع به بیش‌تطبیق می‌کند، هدف شما تغییر می‌کند به بهبود تعمیم از طریق تنظیم مدل. می‌توانید ظرفیت مدل را کاهش دهید، Dropout یا تنظیم وزن‌ها را اضافه کنید و از توقف زودهنگام استفاده کنید. و به‌طور طبیعی، مجموعه‌داده‌ای بزرگ‌تر یا با کیفیت‌تر همیشه بهترین راه برای کمک به تعمیم مدل است.

فرآیند کاری جهانی در یادگیری ماشین

این فصل شامل مباحث زیر است:

* مراحل چارچوب‌بندی یک مسئله‌ی یادگیری ماشین.
* مراحل توسعه‌ی یک مدل عملیاتی.
* مراحل استقرار مدل در محیط تولید و نگهداری از آن.

در مثال‌های قبلی، فرض بر این بود که از قبل یک مجموعه داده‌ی برچسب‌گذاری‌شده در اختیار داشتیم و می‌توانستیم بلافاصله آموزش مدل را شروع کنیم. اما در دنیای واقعی، اغلب این‌گونه نیست. شما کار خود را با یک مجموعه داده آغاز نمی‌کنید، بلکه با یک **مسئله** شروع می‌کنید.

تصور کنید که کسب‌وکار مشاوره‌ی یادگیری ماشین خود را راه‌اندازی کرده‌اید. شرکت خود را ثبت می‌کنید، یک وب‌سایت شیک طراحی می‌کنید و شبکه‌ی خود را مطلع می‌کنید. پروژه‌ها یکی پس از دیگری شروع به سرازیر شدن می‌کنند:

* یک موتور جستجوی عکس شخصی‌سازی‌شده برای یک شبکه اجتماعی اشتراک‌گذاری تصاویر—به گونه‌ای که با تایپ عبارت "عروسی"، تمام عکس‌هایی که در مراسم‌های عروسی گرفته‌اید بدون نیاز به برچسب‌گذاری دستی پیدا شوند.
* شناسایی هرزنامه‌ها و محتوای متنی توهین‌آمیز در میان پست‌های یک اپلیکیشن چت نوپا.
* ساخت یک سیستم پیشنهاد موسیقی برای کاربران یک رادیوی آنلاین.

شناسایی تقلب در کارت‌های اعتباری برای یک وب‌سایت تجارت الکترونیکی.

* پیش‌بینی نرخ کلیک روی تبلیغات نمایشی (Display Ads) برای تصمیم‌گیری در مورد اینکه کدام تبلیغ در چه زمانی به یک کاربر خاص نمایش داده شود.
* شناسایی کوکی‌های غیرعادی روی نوار نقاله‌ی خط تولید کوکی.

استفاده از تصاویر ماهواره‌ای برای پیش‌بینی مکان سایت‌های باستان‌شناسی ناشناخته.  
  
**توجه به اخلاقیات**

ممکن است گاهی پروژه‌هایی به شما پیشنهاد شود که از نظر اخلاقی قابل تردید باشند، مانند «ساخت یک هوش مصنوعی که اعتمادپذیری یک فرد را از روی تصویر چهره‌ی او ارزیابی کند.»  
ابتدا باید گفت اعتبار چنین پروژه‌ای محل تردید است: مشخص نیست چرا اعتمادپذیری باید در چهره‌ی یک فرد منعکس شود.  
ثانیاً، چنین وظیفه‌ای دریچه‌ای به سوی انواع مشکلات اخلاقی باز می‌کند. جمع‌آوری مجموعه داده برای چنین وظیفه‌ای به معنای ثبت تعصبات و پیش‌داوری‌های افرادی است که تصاویر را برچسب‌گذاری می‌کنند. مدل‌هایی که شما بر اساس چنین داده‌هایی آموزش می‌دهید، صرفاً همین تعصبات را در قالب یک الگوریتم جعبه‌سیاه رمزگذاری می‌کنند و به آن‌ها ظاهری از مشروعیت می‌بخشند.

در جامعه‌ای که تا حد زیادی از فناوری بی‌اطلاع است، جمله‌ای مانند «الگوریتم هوش مصنوعی گفت این فرد قابل اعتماد نیست» به طرز عجیبی وزن و اعتبار بیشتری نسبت به جمله‌ی «جان اسمیت گفت این فرد قابل اعتماد نیست» پیدا می‌کند، حتی با وجود اینکه اولی صرفاً یک تخمین یادگرفته‌شده از دومی است. مدل شما در واقع بدترین جنبه‌های قضاوت انسانی را در مقیاس گسترده عملیاتی و مشروع می‌کند، که پیامدهای منفی واقعی بر زندگی افراد دارد.

**فناوری هرگز خنثی نیست.** اگر کار شما هرگونه تأثیری بر جهان دارد، این تأثیر یک جهت اخلاقی دارد: انتخاب‌های فنی نیز انتخاب‌های اخلاقی هستند. همیشه به ارزش‌هایی که می‌خواهید کارتان از آن‌ها حمایت کند، توجه کنید.

بسیار عالی می‌شد اگر می‌توانستید مجموعه داده‌ی صحیح را از **keras.datasets** وارد کرده و بلافاصله شروع به آموزش مدل‌های یادگیری عمیق کنید. متأسفانه، در دنیای واقعی معمولاً باید همه چیز را از ابتدا شروع کنید.

در این فصل، درباره‌ی یک الگوی گام‌به‌گام جهانی خواهید آموخت که می‌توانید از آن برای نزدیک شدن به هر مسئله‌ی یادگیری ماشین و حل آن، مانند مسائل فهرست‌شده‌ی بالا، استفاده کنید. این قالب همه چیزهایی را که در فصل‌های ۴ و ۵ یاد گرفتید، با هم ترکیب می‌کند و به شما یک دیدگاه گسترده‌تر می‌دهد که به شما کمک می‌کند مطالب فصل‌های بعدی را بهتر درک کنید.

**فرآیند کاری جهانی در یادگیری ماشین شامل سه بخش کلی است:**

1. **تعریف وظیفه:** درک حوزه‌ی مسئله و منطق تجاری مرتبط با آنچه مشتری درخواست کرده است. جمع‌آوری یک مجموعه داده، درک اینکه داده‌ها چه چیزی را نشان می‌دهند، و انتخاب نحوه‌ی اندازه‌گیری موفقیت در این وظیفه.
2. **توسعه‌ی مدل:** آماده‌سازی داده‌ها به گونه‌ای که قابل پردازش توسط یک مدل یادگیری ماشین باشند، انتخاب یک پروتکل ارزیابی مدل و یک مرجع ساده برای شکست دادن، آموزش اولین مدل با قدرت تعمیم و توانایی بیش‌تطبیق، سپس تنظیم و بهینه‌سازی مدل تا دستیابی به بهترین عملکرد تعمیم.
3. **استقرار مدل:** ارائه‌ی کار به ذینفعان، ارسال مدل به یک سرور وب، یک اپلیکیشن موبایل، یک صفحه‌ی وب، یا یک دستگاه تعبیه‌شده، و نظارت بر عملکرد مدل.